SerialGrabber Documentation

Release 1

NigelB

December 12, 2016

Contents

We use SerialGrabber to read data from sensor devices, process this data and post it to an internet endpoint.

The data is read using a Readers, cached by a Cache, processed by a Processor, and archived by an Archive see Data Life Cycle.

Contents

CHAPTER 1

Installing SerialGrabber

1.1 Dependencies

SerialGrabber depends on these packages:

- pyserial
- requests

and optionally:

- xbee

The easiest way to install these packages is to use pip:

```
~ $ pip install pyserial requests
.
.
.
```

and optionally:

```
~ $ pip install xbee
.
.
.
```

1.2 From Source

To install Serial Grabber run the following commands as `root`:

```
~ $ git clone https://github.com/nigelb/SerialGrabber.git
~ $ cd SerialGrabber
~/SerialGrabber $ python setup.py install
```

After this you need to create your configuration directory. The easiest way to do this is to copy the `example_config` directory, for example if you wanted to use `/etc/SerialGrabber` as your configuration directory:

```
~/SerialGrabber $ cp -R SerialGrabber/example_config /etc/SerialGrabber
```

Then edit the configuration files and change them according to your needs.

CHAPTER 2

Running SerialGrabber

Once you have installed SerialGravver and created your configuration you can launch SerialGrabber from the command line:

```
#~> serial_grabber --help
usage: serial_grabber [-h] [--config-dir <config_dir>]

Serial Grabber will read the configured serial port and process the data
received.

optional arguments:
  -h, --help            show this help message and exit
  --config-dir <config_dir>
                        The location of the config directory, default:
                        /etc/SerialGrabber


#~> serial_grabber --config-dir /etc/SerialGrabber
```

CHAPTER 3

Data Life Cycle

There are two threads, the Reader thread and the Processor thread.

3.1 Reader Thread

The Reader

1. Reads the data from the configured source, see *reader* and Readers.
2. Most readers use a TransactionExtractor to cut the stream up into discreet payloads.
3. Writes the data into the cache directory, see: *cache_dir*, and Cache

3.2 Processor Thread

The Processor

1. Waits until a cache entry is older than *cache_collision_avoidance_delay*.
2. **Processes the data with the configured processor, see Processor.**
    (a) if the processing was successful the cache entry in moved into the archive, see *archive_dir* and Archive.
    (b) otherwise it is left in the cache and it will be reprocessed in a later cycle.

CHAPTER 4

Settings

4.1 SerialGrabber Settings

Parameters located in the `SerialGrabber_Settings.py` file located in the configuration directory, see: Running SerialGrabber.

4.1.1 cache_collision_avoidance_delay

The minimum amount of time to let a transaction exist in the cache before being processed by the processor.

Important: This avoids the situation where the `processor` thread starts reading the cache entry before the `reader` has finished writing it.

```
cache_collision_avoidance_delay = 1
```

4.1.2 processor_sleep

The amount of time the processor should sleep between iterations.

```
processor_sleep = 1
```

4.1.3 watchdog_sleep

The amount of time that the watchdog thread will sleep for on each iteration.

```
watchdog_sleep = 1
```

4.1.4 reader_error_sleep

The amount of time the reader thread will sleep if there is an error. This avoids the reader's thread using all of the cpu when there is an error.

```
reader_error_sleep = 1
```

4.1.5 startup_ignore_threshold_milliseconds

The amount of time that the reader will ignore input from the reader on startup. Some devices spew some garbage characters out on the serial port when they are powered up. This option helps avoid mistaking this garbage for real data.

```
startup_ignore_threshold_milliseconds = 1000
```

4.1.6 drop_carriage_return

If this option is *True* then `\r (0x0A)` characters are removed from the data stream.

4.1.7 reader

An object that implements `serial_grabber.reader.Reader`, see Readers

```
reader = SerialReader(1000, '/dev/ttyUSB0', 115200,
    timeout=1,
    parity=serial.PARITY_NONE,
    stop_bits=1)
```

4.1.8 processor

An object that implements `serial_grabber.processor.Processor`, see: Processor

```
from serial_grabber.processor.UploadProcessor import UploadProcessor

processor = UploadProcessor("https://example.org/cgi-bin/data.py", form_params={'device':'Device-1'})
```

4.1.9 Example Config

```
#!/usr/bin/env python
# SerialGrabber reads data from a serial port and processes it with the
# configured processor.
# Copyright (C) 2012  NigelB
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along
# with this program; if not, write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

import serial
from serial_grabber.reader import TransactionExtractor
from serial_grabber.reader.FileReader import FileReader
```

```
from serial_grabber.reader.SerialReader import SerialReader
from serial_grabber.processor.UploadProcessor import UploadProcessor

#Serial Settings
timeout = 1
port = "/dev/ttyUSB0"
baud = 57600
parity = serial.PARITY_NONE
stop_bits = 1

#Settings
cache_collision_avoidance_delay = 1
processor_sleep = 1
watchdog_sleep = 1

reader_error_sleep = 1

drop_carriage_return = True

transaction = TransactionExtractor("default", "BEGIN DATA", "END DATA")

reader = SerialReader(transaction,
                      1000,
                      port,
                      baud,
                      timeout=timeout,
                      parity=parity,
                      stop_bits=stop_bits)

processor = UploadProcessor("https://example.org/cgi-bin/upload.py")
```

4.2 SerialGrabber Paths

Parameters located in the `SerialGrabber_Paths.py` file located in the configuration directory, see: Running SerialGrabber.

4.2.1 cache_dir

The location of the cache directory, see Data Life Cycle

```
cache_dir = "/path/to/cache/directory"
```

4.2.2 archive_dir

The location of the archive directory, see Data Life Cycle

```
archive_dir = "/path/to/archive/directory"
```

4.2.3 Example Config

```
#!/usr/bin/env python
# SerialGrabber reads data from a serial port and processes it with the
# configured processor.
# Copyright (C) 2012  NigelB
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along
# with this program; if not, write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

import os

#Directories
data_logger_dir = None
if "APPDATA" in os.environ:
    data_logger_dir = os.path.join(os.environ["APPDATA"], "datalogger")
else:
    data_logger_dir = os.path.join(os.path.expanduser("~"), ".datalogger")

cache_dir = os.path.join(data_logger_dir, "cache")
archive_dir = os.path.join(data_logger_dir, "archive")
```

4.3 SerialGrabber Storage

Parameters located in the `SerialGrabber_Storage.py` file located in the configuration directory, see: Running SerialGrabber.

4.3.1 Example Config

For documentation on the classes below see: Cache and Archive.

```
#!/usr/bin/env python
# SerialGrabber reads data from a serial port and processes it with the
# configured processor.
# Copyright (C) 2012  NigelB
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along
# with this program; if not, write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

from serial_grabber.archive import *
from serial_grabber.cache import FileSystemCache
from serial_grabber.util import RollingFilename, PreviousWeekStartBoundary, week_period, to_date_forr


storage_archive = JSONLineArchive(
    SerialGrabber_Paths.archive_dir,
    RollingFilename(
        PreviousWeekStartBoundary(),
        week_period,
        None,
        to_date_format
    )
)

storage_cache = FileSystemCache(SerialGrabber_Paths.cache_dir, storage_archive)
```

CHAPTER 5

Readers

5.1 FileReader

class serial_grabber.reader.FileReader.**FileReader**(*transaction_extractor*, *filename*)

A reader that opens and reads a file for its input.

Parameters

- **transaction_extractor** (serial.grabber.reader.TransactionExtractor) – The transaction extractor used to parse the input stream.
- **filename** (*str*) – The file to open and read as input.

```
from serial_grabber.reader.FileReader import FileReader

...

transaction = TransactionExtractor("default", "BEGIN DATA", "END DATA")
reader = FileReader(transaction, "data.txt")
```

5.2 SerialReader

class serial_grabber.reader.SerialReader.**SerialReader**(*transaction_extractor*, *startup_ignore_threshold_milliseconds*, *serial_connection*)

A reader that connects to the specified serial port for its input.

Parameters

- **transaction_extractor** (serial.grabber.reader.TransactionExtractor) – The transaction extractor used to parse the input stream.
- **startup_ignore_threshold_milliseconds** (*int*) – The interval that input is ignored for at startup

```
from serial_grabber.reader.SerialReader import SerialReader

...

transaction = TransactionExtractor("default", "BEGIN DATA", "END DATA")
reader = SerialReader(transaction,
                      '/dev/ttyUSB0',
```

```
                        115200,
                        timeout=1,
                        parity=serial.PARITY_NONE,
                        stop_bits=1)
```

5.3 TCPReader

```
from serial_grabber.reader.TCP import TCPReader

...

transaction = TransactionExtractor("default", "BEGIN DATA", "END DATA")
reader = TCPReader(transaction, "example.org", 8111)
```

5.4 PacketRadioReader

class serial_grabber.reader.Xbee.**PacketRadioReader**(*serial_connection*, *radio_class=<class 'ZigBee'>*, *packet_filter=<function <lambda>>*, ***kwargs*)

Reads Digi Xbee/ZigBee API mode packets from the configured serial port

Parameters

- **port** (*str*) – The serial port to use, eg: /dev/ttyUSB0
- **baud** (*int*) – The baud rate to use, eg: 115200
- **timeout** (*int*) – eg: 60
- **parity** (*int*) – eg: serial.PARITY_NONE
- **stop_bits** (*int*) – eg: serial.STOPBITS_ONE
- **radio_class** (*xbee.base.XBeeBase*) – The implementation to use, eg: xbee.zigbee.ZigBee
- **packet_filter** (*lambda a: True*) – A function that takes one parameter, the parsed radio packet, and returns a bool specifying weather or not to keep the packet.
- **escaped** (*bool*) – The radio is in API mode 2

```
import serial
from serial_grabber.reader.Xbee import PacketRadioReader

...

reader = DigiRadioReader("/dev/ttyUSB0", 115200,
                         timeout=60,
                         parity=serial.PARITY_NONE,
                         stop_bits=serial.STOPBITS_ONE,
                         packet_filter=lambda packet: packet['id'] == 'rx',
                         escaped=True)
```

5.5 StreamRadioReader

class serial_grabber.reader.Xbee.**StreamRadioReader**(*stream_transaction_factory*, *serial_connection*, *message_verifier=<serial_grabber.reader.Xbee.MessageVerifier instance>*, *radio_class=<class 'ZigBee'>*, *packet_filter=<function <lambda>>*, *binary=True*, ***kwargs*)

Reads Digi Xbee/ZigBee API mode packets from the configured serial port, converts the packets into a stream for each MAC Address and passes the stream onto a `serial.grabber.reader.TransactionExtractor`

Parameters

- **stream_transaction_factory** (*fn(stream_id)*) – The function that creates a `serial.grabber.reader.TransactionExtractor` with the specified stream_id
- **serial_connection** (*SerialConnection*) – A serial connection object
- **radio_class** (*xbee.base.XBeeBase*) – The implementation to use, eg: xbee.zigbee.ZigBee
- **packet_filter** (*lambda a: True*) – A function that takes one parameter, the parsed radio packet, and returns a bool specifying weather or not to keep the packet.
- **binary** – Weather the data recieved needs to be base64 encoded by the cache (otherwise binary data may mess up the cache json)
- **escaped** (*bool*) – The radio is in API mode 2

```
import serial
from serial_grabber.reader.Xbee import StreamRadioReader

def create_stream(stream_id):
    print " ".join([format(ord(x), "02x") for x in stream_id])
    return TransactionExtractor(stream_id, start_del, end_del)

reader = DigiRadioReader(create_stream,
                         "/dev/ttyUSB0",
                         115200,
                         timeout=60,
                         parity=serial.PARITY_NONE,
                         stop_bits=serial.STOPBITS_ONE,
                         packet_filter=lambda packet: packet['id'] == 'rx',
                         escaped=True)
```

CHAPTER 6

TransactionExtractor

CHAPTER 7

Cache

Once data has acquired by the Readers and cut into a transaction (see Data Life Cycle), the payload is inserted into the cache.

The Cache settings can be configure in: SerialGrabber Storage.

7.1 FileSystemCache

class serial_grabber.cache.**FileSystemCache**(*cache_dir*, *archive*)

An implementation of serial_grabber.cache.Cache that stores the cache entries on the local filesystem.

Parameters

- **cache_dir** (*str*) – The directory location to store the cached entries.
- **archive** (serial_grabber.archive.BaseArchive) – The archive implementation used to archive the processed data.

CHAPTER 8

Processor

8.1 CompositeProcessor

class serial_grabber.processor.**CompositeProcessor**(*processors=()*, *composition_operation=<function <lambda>>*, *starting_value=False*)

Allows processing by multiple processors.

Parameters

- **processors** (*List of objects that implement serial_grabber.processor.Processor*) – The list of Processors.
- **composition_operation** (*lambda a, b*) – The function that composes the Processor results, default: `lambda a, b: a or b`.
- **starting_value** (*bool*) – The initial value passed to the composition_operation with the result from the first processor.

8.2 CSVFileProcessor

class serial_grabber.processor.CSVProcessors.**CSVFileProcessor**(*filename=None*, *permission=420*, *headers=None*)

Converts the payload to a CSV file and appends it to *filename*

Parameters

- **filename** (*string*) – The output location
- **permission** (*int*) – The file mode to set on the output file

8.3 FileAppenderProcessor

class serial_grabber.processor.FileAppenderProcessor.**FileAppenderProcessor**(*output_file*, *entry_delimiter='n'*)

Append the contents of the transactions payload to *output_file*.

Parameters

- **output_file** (*string*) – The name of the file to append to.
- **entry_delimiter** (*string*) – This value will be written to the file after each payload. None will disable this function.

8.4 JsonFileProcessor

class serial_grabber.processor.JsonFileProcessor.**JsonFileProcessor**(*output_file*, *transaction_filter=None*, *limit=-1*, *permission=420*)

Writes the last *limit* transactions that were not filtered out by *transaction_filter* as a JSON encoded array to *output_file*.

Parameters

- **output_file** (*str*) – The filename of the json output file.
- **transaction_filter** (*serial_grabber.filter.TransactionFilter or None*) – Used to filter transactions.
- **limit** (*int*) – The number of transactions keep and write to the output_file, -1 for unlimited.
- **permission** (*int*) – The file permissions to set on the output_file.

8.5 RollingFilenameProcessor

class serial_grabber.processor.**RollingFilenameProcessor**(*boundary*, *period_ms*, *output_dir*, *file_extension*, *output_processor*)

Used to change the output filename of processors that implement `serial_grabber.processor.ExternalFilenameProcessor`. The file names are aligned on *boundary* it rolled forward every *period_ms* with a call to *output_processor.setOutputFileName* having the form of: *output_dir/date_time.file_extension*.

Parameters

- **boundary** (*int*) – The boundary on which to align the filename roll on.
- **period_ms** (*int*) – The period (in milliseconds) to change the filename on.
- **output_dir** (*string*) – The directory to write the output to
- **file_extension** (*string*) – The file extension to give the output file.
- **output_processor** (*serial_grabber.processor.ExternalFilenameProcessor*) – The processor to process the chunk.

```
from serial_grabber.processor import RollingFilenameProcessor

processor = RollingFilenameProcessor(
    PreviousMidnightBoundary(), 60 * 60 * 1000,
    "/home/user/data/aquarium/10_sec",
    "csv",
```

```
    CSVFileProcessor()
)
```

8.6 TransformProcessor

class serial_grabber.processor.**TransformProcessor**(*transform*, *processor*)

Transforms the transaction being processed and passes it to the specified Processor.

Parameters

- **transform** (*serial_grabber.transform.Transform*) – The transformation to use
- **processor** (*serial_grabber.processor.Processor*) – The Processor to pass the transformed transaction

8.7 UploadProcessor

class serial_grabber.processor.UploadProcessor.**UploadProcessor**(*url*, *upload_error_sleep=10*, *auth=None*, *form_params=None*, *headers=None*, *request_kw=None*, *success_http_codes=[200]*, *reject_http_codes=[406]*, *format_url=False*)

Encodes the data as a HTTP form and uploads it to the configured url.

Parameters

- **url** (*str*) – The url to upload the data to.
- **upload_error_sleep** (*int*) – The amount of time to sleep if an error occurred during upload.
- **auth** (*serial_grabber.processor.UploadProcessor.HTTPBasicAuthentication or None*) – The credentials to use.
- **form_params** (*dict*) – Addition parameters to be added to the uploaded form.
- **headers** (*dict*) – Headers to add to the upload requests.
- **request_kw** (*dict*) – Parameters that are passed to the requests.post method call
- **success_http_code** (*dict*) – A list of HTTP status codes indicating that the server accepted the data, this data will be moved into the archive.
- **reject_http_code** (*dict*) – A list of HTTP status codes indicating that the server rejected the data, this data will be moved into the bad data archive.
- **format_url** (*dict*) – If True then in the processing method the url will be set to: `url.format(**process_entry[constants.url_parameters])` which should be set by your transform.

HTTP Basic Authentication:

```
from serial_grabber.processor.UploadProcessor import UploadProcessor
from requests.auth import HTTPBasicAuth

processor = UploadProcessor("https://example.org/cgi-bin/data.py",
    auth=HTTPBasicAuth("username", "password"),
    form_params={'device':'Device-1'}
)
```

HTTP Digest Authentication with SSL verification disabled:

```
from serial_grabber.processor.UploadProcessor import UploadProcessor
from requests.auth import HTTPDigestAuth

processor = UploadProcessor("https://example.org/cgi-bin/data.py",
    auth=HTTPDigestAuth("username", "password"),
    form_params={'device':'Device-1'},
    request_kw={'verify':False}
)
```

CHAPTER 9

Archive

Once a payload has been successfully processed by the configured processor (see Data Life Cycle) the payload item is removed from the cache and sent to the configured archive.

The Archive settings can be configure in: SerialGrabber Storage.

9.1 FileSystemArchive

class serial_grabber.archive.**FileSystemArchive**(*archive_dir*)

A serial_grabber.archive.BaseArchive implementation that moves the cache entry from the cache directory to the archive directory.

Parameters **archive_dir** (*str*) – The directory in which to store the archive files.

9.2 DumpArchive

class serial_grabber.archive.**DumpArchive**(*archive_dir*)

A serial_grabber.archive.BaseArchive implementation that deletes the cache entry data from the cache.

9.3 JSONLineArchive

class serial_grabber.archive.**JSONLineArchive**(*archive_dir*, *filename_roller=None*)

A serial_grabber.archive.BaseArchive implementation that stores each cache entry as a JSON encoded line within the archive file.

Parameters

- **archive_dir** (*str*) – The directory in which to store the archive files.
- **filename_roller** (serial_grabber.util.RollingFilename or None) – The filename roller to use.

CHAPTER 10

Complex Examples

The following example are taken from some of our deployments.

10.1 Aquarium Example

An Arduino mega is being used to control the PH of an aquarium by injecting CO_2 into the water. The Arduino outputs data on its serial port so that it can be captured and displayed in real time.

10.1.1 Sample Data

```
BEGIN AQUARIUM
temp,25.06,20.00
ph,3.49,7.00
temp_pid,1.0000000000,-0.3000000000,-1.0000000000,255.0000000000
ph_pid,1.0000000000,1.0000000000,1.0000000000,736.9911499023
END AQUARIUM
BEGIN AQUARIUM
temp,25.06,20.00
ph,3.49,7.00
temp_pid,1.0000000000,-0.3000000000,-1.0000000000,255.0000000000
ph_pid,1.0000000000,1.0000000000,1.0000000000,744.0111694335
END AQUARIUM
```

10.1.2 Configuration

SerialGrabber_Settings.py

```
import serial
from serial_grabber.reader import TransactionExtractor
from serial_grabber.reader.FileReader import FileReader
from serial_grabber.reader.SerialReader import SerialReader
from serial_grabber.processor.UploadProcessor import UploadProcessor

#Serial Settings
timeout = 1
port = "/dev/ttyUSB0"
baud = 115200
parity = serial.PARITY_NONE
```

```
stop_bits = 1

#Settings
cache_collision_avoidance_delay = 1
processor_sleep = 1
watchdog_sleep = 1

reader_error_sleep = 1

drop_carriage_return = True

transaction = TransactionExtractor("default", "BEGIN AQUARIUM", "END AQUARIUM")

reader = SerialReader(transaction,
                      1000,
                      port,
                      baud,
                      timeout=timeout,
                      parity=parity,
                      stop_bits=stop_bits)

processor = CompositeProcessor([
    FileAppenderProcessor("all.txt"),
    TransformProcessor(AquariumTransform(), CompositeProcessor([
        JsonFileProcessor("data/processed/current.json", None, 1),
        TransformProcessor(BlockAveragingTransform(10, averageAquariumData),
            RollingFilenameProcessor(PreviousMidnightBoundary(), 60 * 60 * 1000, "/home/user/data/aqu

        TransformProcessor(BlockAveragingTransform(10 * 60, averageAquariumData),
            RollingFilenameProcessor(PreviousMidnightBoundary(), 24 * 60 * 60 * 1000, "/home/user/dat

        TransformProcessor(BlockAveragingTransform(60 * 60, averageAquariumData),
            RollingFilenameProcessor(PreviousWeekStartBoundary(), 7 * 24 * 60 * 60 * 1000, "/home/use
    ])),


])
```

10.2 Eco Fest Example

10.2.1 Configuration

SerialGrabber_Settings.py

```
processor = CompositeProcessor([
    FileAppenderProcessor("/home/user/data/eco/all.txt"),
    TransformProcessor(EcoFestTransform(), CompositeProcessor([
        JsonFileProcessor("/home/user/data/eco/every_10.json", CountingTransactionFilter(10), 72),
        JsonFileProcessor("/home/user/data/eco/current.json", None, 1)]))
])
```

10.3 ThingSpeak Example

10.3.1 Sample Data

```
BEGIN SENSORS
1,964
2,50
3,513
4,645
5,87
6,875
7,365
8,798
END SENSORS
```

10.3.2 Configuration

```
#!/usr/bin/env python
# SerialGrabber reads data from a serial port and processes it with the
# configured processor.
# Copyright (C) 2012  Nigel B
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along
# with this program; if not, write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

import serial

from serial_grabber.extractors import TransactionExtractor
from serial_grabber.reader.FileReader import FileReader
from serial_grabber.processor import TransformProcessor
from serial_grabber.processor.UploadProcessor import UploadProcessor
from ThingSpeakTransformer import ThingSpeakTransformer

# Serial Settings
timeout = 1
port = "/dev/ttyUSB0"
baud = 57600
parity = serial.PARITY_NONE
stop_bits = 1

# Settings
cache_collision_avoidance_delay = 1
processor_sleep = 1
watchdog_sleep = 1
```

```
reader_error_sleep = 1

drop_carriage_return = True

transaction = TransactionExtractor("default", "BEGIN SENSORS", "END SENSORS")

reader = FileReader(transaction, "ThingSpeakExample.data")

uploader = UploadProcessor("http://api.thingspeak.com/update",
                           headers={'content-type': 'application/x-www-form-urlencoded'},
                           )

processor = TransformProcessor(ThingSpeakTransformer(), uploader)
```

10.3.3 Transformer

```
#!/usr/bin/env python
# SerialGrabber reads data from a serial port and processes it with the
# configured processor.
# Copyright (C) 2012  Nigel B
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along
# with this program; if not, write to the Free Software Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
from serial_grabber import constants
from serial_grabber.transform import Transform


api_key = "GT14ED5QGNLE2E8N"

field_map = {
    "1": "field8",
    "2": "field7",
    "3": "field6",
    "4": "field5",
    "5": "field4",
    "6": "field3",
    "7": "field2",
    "8": "field1",
}

# A Transform is used by a TransformProcessor to transform the data received
# from the reader into a form that can be used for the configured processor.
class ThingSpeakTransformer(Transform):

    def transform(self, process_entry):
```

```
        # Retrieve the transaction's data from the process_entry
        payload = process_entry[constants.data][constants.payload]

        transformed_data = {"api_key": api_key}

        # Process the retrieved data into
        lines = payload.split("\n")
        for line in lines:

            # Strip out the start and end delimiters
            if "SENSORS" in line:
                continue

            # Extract the sensor ID and value from the line
            sensor_id, sensor_value = line.split(",")
            sensor_value = sensor_value

            transformed_data[field_map[sensor_id]] = sensor_value

        process_entry[constants.data][constants.payload] = transformed_data
        return process_entry
```

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

C

D

F

J

P

R

S

T

U